AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method for performing a lock-free update to one or more fields in an existing node in a linked list, comprising:

receiving a reference to the existing node in the linked list, wherein the existing node contains the one or more fields to be updated;

obtaining a new node to be added to the linked list, wherein other processes do not possess references to the new node and therefore cannot initially access the new node;

copying a snapshot of the existing node to the new node, which includes copying a next pointer of the existing node to the new node, so that the new node points to a node immediately following the existing node;

updating one or more fields in the new node that correspond to the one or more fields in the existing node;~~ and~~

performing a single atomic operation that modifies the next pointer of the existing node to point to the new node and also marks the next pointer to indicate that the existing node is deleted, ~~wherein the existing node remains in the linked list after being marked as deleted,~~ whereby the new node becomes part of the linked list and the existing node is deleted in a single atomic operation; and

splicing the existing node out of the linked list by atomically modifying the next pointer of a node immediately preceding the existing node in the linked list to point to the new node, instead of pointing to the existing node.

2. (Cancelled)

3. (Currently amended) The method of claim 1~~2~~, wherein if a process that deleted the existing node does not perform the splicing operation, another process, which subsequently detects that the existing node has been deleted, performs the splicing operation.

4. (Currently amended) The method of claim 1, wherein copying a snapshot of the existing node to the new node involves:

copying the contents of the existing node to the new node;

examining the next pointer of the existing node to determine if the existing node has been deleted; and

if so ~~the existing node has been deleted~~, taking a remedial action~~.~~;

otherwise, not taking the remedial action.

5. (Currently amended) The method of claim 4, wherein taking the remedial action involves:

following the next pointer of the existing node in an attempt to find an updated version of the existing node;~~and~~

if an updated version of the existing node is found, copying a snapshot of the updated version of the existing node to the new node~~.~~; and

if an updated version of the existing node is not found, indicating that the remedial action fails.

6. (Original) The method of claim 1, further comprising deleting a target node from the linked list by:

receiving a reference to the target node to be deleted from the linked list;

atomically marking a next pointer in the target node to indicate that the target node is deleted; and

atomically modifying the next pointer of a node immediately preceding the target node in the linked list to point to a node immediately following the target node in the linked list, instead of pointing to the target node, thereby splicing the target node out of the linked list.

7. (Original) The method of claim 6, wherein after the target node is spliced out of the linked list, the method further comprises modifying the next pointer of the target node so that the next pointer remains marked but points to a node immediately preceding the target node instead of the node immediately following node the target node in the linked list.

8. (Original) The method of claim 1, further comprising inserting an additional node into the linked list by:

identifying a node immediately preceding the additional node in the linked list;

identifying a node immediately following the additional node in the linked list; and

splicing the additional node into the linked list by,

setting the next pointer for the additional node to point to the immediately following node, and

atomically updating the next pointer of the immediately preceding node to point to the additional node.

9. (Currently amended) The method of claim 1, further comprising reading a snapshot of multiple fields from a target node in the linked list by:

reading the multiple fields from the target node;

examining the next pointer of the target node to determine if the target node has been deleted; and

if so, ~~the target node has been deleted,~~ taking a remedial action~~.~~;

otherwise, not taking the remedial action.

10. (Currently amended) The method of claim 9, wherein taking the remedial action involves:

following the next pointer of the target node in an attempt to find an updated version of the target node; ~~and~~

if an updated version of the target node is found, repeating the process of reading a snapshot of the multiple fields from the updated version of the target node~~.~~; and

if an updated version of the existing node is not found, indicating that the remedial action fails.

11. (Original) The method of claim 1, wherein atomically modifying the next pointer of the existing node to indicate that the existing node is deleted involves setting a "deleted bit" in the next pointer.

12. (Currently amended) The method of claim 1, wherein while atomically modifying the next pointer of the existing node,

if the next pointer indicates that the existing node is already deleted, the atomic modification operation fails and the method further comprises taking a remedial action to deal with the fact that the existing node is already deleted;~~.~~

otherwise, continuing performing the atomic modification operation.

13. (Original) The method of claim 1, wherein a given node in the linked list includes:

a key that contains an identifier for the given node;

one or more fields containing data values or pointers to data values associated with the given node; and

a next pointer that contains the address of a node that immediately follows the given node in the linked list, and that also contains a deleted indicator, which indicates whether the given node has been deleted.

14. (Original) The method of claim 1, further comprising periodically performing a garbage-collection operation to reclaim deleted nodes that have become unreachable.

15. (Currently amended) A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method for performing a lock-free update to one or more fields in an existing node in a linked list, the method comprising:

receiving a reference to the existing node in the linked list, wherein the existing node contains the one or more fields to be updated;

obtaining a new node to be added to the linked list, wherein other processes do not possess references to the new node and therefore cannot initially access the new node;

copying a snapshot of the existing node to the new node, which includes copying a next pointer of the existing node to the new node so that the new node points to a node immediately following the existing node;

updating one or more fields in the new node that correspond to the one or more fields in the existing node; ~~and~~

performing a single atomic operation that modifies a next pointer of the existing node to point to the new node and also marks the next pointer to indicate that the existing node is deleted, ~~wherein the existing node remains in the linked list after being marked as deleted,~~ whereby the new node becomes part of the linked list and the existing node is deleted in a single atomic operation; and

splicing the existing node out of the linked list by atomically modifying the next pointer of a node immediately preceding the existing node in the linked list to point to the new node, instead of pointing to the existing node.

16. (Cancelled)

17. (Currently amended) The computer-readable storage medium of claim 15 ~~16~~, wherein if a process that deleted the existing node does not perform the splicing operation, another process, which subsequently detects that the existing node has been deleted, performs the splicing operation.

18. (Currently amended) The computer-readable storage medium of claim 15, wherein copying the snapshot of the existing node to the new node involves:

copying the contents of the existing node to the new node;

examining the next pointer of the existing node to determine if the existing node has been deleted; and

if so, ~~the existing node has been deleted,~~ taking a remedial action~~.~~;

otherwise, not taking the remedial action.

19. (Currently amended) The computer-readable storage medium of claim 18, wherein taking the remedial action involves:

following the next pointer of the existing node in an attempt to find an updated version of the existing node; ~~and~~

if an updated version of the existing node is found, copying a snapshot of the updated version of the existing node to the new node~~.~~; and

if an updated version of the existing node is not found, indicating that the remedial action fails.

20. (Original) The computer-readable storage medium of claim 15, wherein the method further comprises deleting a target node from the linked list by:

receiving a reference to the target node to be deleted from the linked list;

atomically marking a next pointer in the target node to indicate that the target node is deleted; and

atomically modifying the next pointer of a node immediately preceding the target node in the linked list to point to a node immediately following the target node in the linked list, instead of pointing to the target node, thereby splicing the target node out of the linked list.

21. (Original) The computer-readable storage medium of claim 20, wherein after the target node is spliced out of the linked list, the method further comprises modifying the next pointer of the target node so that the next pointer remains marked but points to a node immediately preceding the target node instead of the node immediately following node the target node in the linked list.

22. (Original) The computer-readable storage medium of claim 15, wherein the method further comprises inserting an additional node into the linked list by:

identifying a node immediately preceding the additional node in the linked list;

identifying a node immediately following the additional node in the linked list; and

splicing the additional node into the linked list by,

setting the next pointer for the additional node to point to the immediately following node, and

atomically updating the next pointer of the immediately preceding node to point to the additional node.

23. (Currently amended) The computer-readable storage medium of claim 15, wherein the method further comprises reading a snapshot of multiple fields from a target node in the linked list by:

reading the multiple fields from the target node;

examining the next pointer of the target node to determine if the target node has been deleted; and

if so, ~~the target node has been deleted,~~ taking a remedial action~~.~~;

otherwise, not taking the remedial action.

24. (Currently amended) The computer-readable storage medium of claim 23, wherein taking the remedial action involves:

following the next pointer of the target node in an attempt to find an updated version of the target node; ~~and~~

if an updated version of the target node is found, repeating the process of reading a snapshot of the multiple fields from the updated version of the target node~~.~~; and

if an updated version of the existing node is not found, indicating that the remedial action fails.

25. (Original) The computer-readable storage medium of claim 15, wherein atomically modifying the next pointer of the existing node to indicate that the existing node is deleted involves setting a "deleted bit" in the next pointer.

26. (Currently amended) The computer-readable storage medium of claim 15, wherein while atomically modifying the next pointer of the existing node,

if the next pointer indicates that the existing node is already deleted, the atomic modification operation fails and the method further comprises taking a remedial action to deal with the fact that the existing node is already deleted~~.~~;

otherwise, continuing performing the atomic modification operation.

27. (Original) The computer-readable storage medium of claim 15, wherein a given node in the linked list includes:

a key that contains an identifier for the given node;

one or more fields containing data values or pointers to data values associated with the given node; and

a next pointer that contains the address of a node that immediately follows the given node in the linked list, and that also contains a deleted indicator, which indicates whether the given node has been deleted.

28. (Original) The computer-readable storage medium of claim 15, wherein the method further comprises periodically performing a garbage-collection operation to reclaim deleted nodes that have become unreachable.

29. (Currently amended) An apparatus that performs a lock-free update to one or more fields in an existing node in a linked list, comprising:

a node obtaining mechanism configured to obtain a new node to be added to the linked list, wherein other processes do not possess references to the new node and therefore cannot initially access the new node;

a copying mechanism configured to copy a snapshot of the existing node to the new node, which includes copying a next pointer of the existing node to the new node so that the new node points to a node immediately following the existing node;

an updating mechanism configured to update one or more fields in the new node that correspond to the one or more fields in the existing node; ~~and~~

a modification mechanism configured to perform a single atomic operation that modifies a next pointer of the existing node to point to the new node and also marks the next pointer to indicate that the existing node is deleted, ~~wherein the existing node remains in the linked list after being marked as deleted,~~ whereby the new node becomes part of the linked list and the existing node is deleted in a single atomic operation; and

a splicing mechanism configured to splice the existing node out of the linked list by atomically modifying the next pointer of a node immediately preceding the existing node in the linked list to point to the new node, instead of pointing to the existing node.

30. (Cancelled)

31. (Currently amended) The apparatus of claim 29 ~~30~~, wherein if a process that deleted the existing node does not activate the splicing mechanism, another process, which subsequently detects that the existing node has been deleted, activates the splicing mechanism.

32. (Currently amended) The apparatus of claim 29, wherein the copying mechanism is configured to:

copy the contents of the existing node to the new node;

examine the next pointer of the existing node to determine if the existing node has been deleted; and

if so, ~~the existing node has been deleted,~~ to take a remedial action~~.~~;

otherwise, not taking the remedial action.

33. (Currently amended) The apparatus of claim 32, wherein while taking the remedial action, the copying mechanism is configured to:

follow the next pointer of the existing node in an attempt to find an updated version of the existing node; ~~and~~

if an updated version of the existing node is found, to copy a snapshot of the updated version of the existing node to the new node~~.~~; and

if an updated version of the existing node is not found, indicating that the remedial action fails.

34. (Original) The apparatus of claim 29, further comprising a deletion mechanism configured to delete a target node from the linked list, wherein the deletion mechanism is configured to:

receive a reference to the target node to be deleted from the linked list;

atomically mark a next pointer in the target node to indicate that the target node is deleted; and to

atomically modify the next pointer of a node immediately preceding the target node in the linked list to point to a node immediately following the target node in the linked list, instead of pointing to the target node, thereby splicing the target node out of the linked list.

35. (Original) The apparatus of claim 34, wherein after the target node is spliced out of the linked list, the deletion mechanism is configured to modify the next pointer of the target node so that the next pointer remains marked but points to a node immediately preceding the target node instead of the node immediately following node the target node in the linked list.

36. (Original) The apparatus of claim 29, further comprising an insertion mechanism configured to insert an additional node into the linked list, wherein the insertion mechanism is configured to:

identify a node immediately preceding the additional node in the linked list;

identify a node immediately following the additional node in the linked list; and to

splice the additional node into the linked list by,

setting the next pointer for the additional node to point to the immediately following node, and

atomically updating the next pointer of the immediately preceding node to point to the additional node.

37. (Currently amended) The apparatus of claim 29, further comprising a reading mechanism configured to read a snapshot of multiple fields from a target node in the linked list, wherein the reading mechanism is configured to:

read the multiple fields from the target node;

examine the next pointer of the target node to determine if the target node has been deleted; and

if so, ~~the target node has been deleted,~~ to take a remedial action~~.~~;

otherwise, not taking the remedial action.

38. (Currently amended) The apparatus of claim 37, wherein while taking the remedial action, the reading mechanism is configured to:

follow the next pointer of the target node in an attempt to find an updated version of the target node; ~~and~~

if an updated version of the target node is found, to repeat the process of reading a snapshot of the multiple fields from the updated version of the target node~~.~~; and

if an updated version of the existing node is not found, indicating that the remedial action fails.

39. (Original) The apparatus of claim 29, wherein while atomically modifying the next pointer of the existing node to indicate that the existing node is deleted, the modification mechanism is configured to set a "deleted bit" in the next pointer.

40. (Currently Amended) The apparatus of claim 29, wherein while atomically modifying the next pointer of the existing node,

if the next pointer indicates that the existing node is already deleted, the modification mechanism is configured to:

fail the modification operation fails; and to

take a remedial action to deal with the fact that the existing node is already deleted~~.~~;

otherwise, continue performing the atomic modification operation.

41. (Original) The apparatus of claim 29, wherein a given node in the linked list includes:

a key that contains an identifier for the given node;

one or more fields containing data values or pointers to data values associated with the given node; and

a next pointer that contains the address of a node that immediately follows the given node in the linked list, and that also contains a deleted indicator, which indicates whether the given node has been deleted.

42. (Original) The apparatus of claim 29, further comprising a garbage collection mechanism configured to periodically perform a garbage-collection operation to reclaim deleted nodes that have become unreachable.